

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет о лабораторной работе №15 по дисциплине основы программной
инженерии**

Выполнил:

Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

1. Декораторы функций в языке Python

Примеры из методических указаний

```
e1.py x e2.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      def higher_order(func):
6          print('Получена функция {} в качестве аргумента'.format(func))
7          func()
8          return func
9
10
11     def wrapper_function():
12         def hello_world():
13             print('Hello world!')
14         hello_world()
15
16
17     def hello_world():
18         print('Hello world!')
19
20
21     ▶ if __name__ == "__main__":
22         print(type(hello_world))
23
24
25         class Hello:
26
27             pass
28
29
30         print(type(Hello))
31         print(type(10))
32         hello = hello_world
33         hello()
34         wrapper_function()
35         print(higher_order(hello_world))
36
```

```
e1 x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe C:/Users/Evil/PycharmPro
<class 'function'>
<class 'type'>
<class 'int'>
Hello world!
Hello world!
Получена функция <function hello_world at 0x000001947735D1F0> в качестве аргумента
Hello world!
<function hello_world at 0x000001947735D1F0>

Process finished with exit code 0
```

Как работают декораторы

```
e1.py x e2.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def decorator_function(func):
6      def wrapper():
7          print('Функция-обёртка!')
8          print('Оборачиваемая функция: {}'.format(func))
9          print('Выполняем обёрнутую функцию...')
10         func()
11         print('Выходим из обёртки')
12     return wrapper
13
14
15  @decorator_function
16  def hello_world():
17      print('Hello world!')
18
19
20  ▶  if __name__ == "__main__":
21      hello_world()
22
```

```
e2 x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe C:/Users/Ev
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x0000002245809D310>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

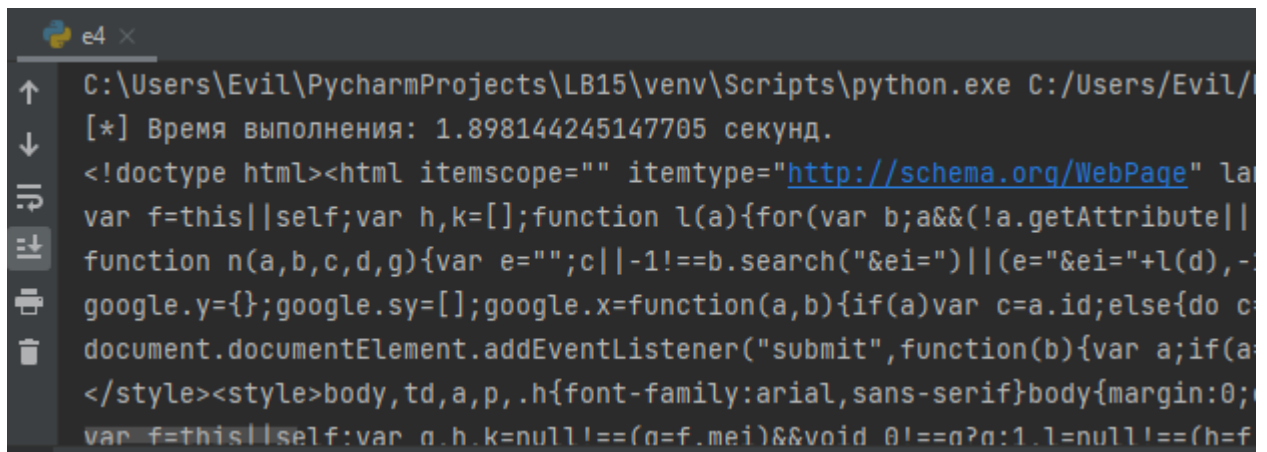
```
e1.py x e2.py x e3.py x
1  ▶  1  #!/usr/bin/env python3
2      2  # -*- coding: utf-8 -*-
3
4
5  5  def benchmark(func):
6      6      import time
7
8      8      def wrapper():
9          9          start = time.time()
10         10         func()
11         11         end = time.time()
12         12         print('[*] Время выполнения: {} секунд.'.format(end - start))
13
14         14         return wrapper
15
16
17  17  @benchmark
18  18  def fetch_webpage():
19      19      import requests
20      20      webpage = requests.get('https://google.com')
21
22
23  23  if __name__ == "__main__":
24      24      fetch_webpage()
25
```

```
e3 x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\p
[*] Время выполнения: 1.2031803131103516 секунд.

Process finished with exit code 0
```

Используем аргументы и возвращаем значения

```
e1.py x e2.py x e3.py x e4.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def benchmark(func):
6     import time
7
8     def wrapper(*args, **kwargs):
9         start = time.time()
10        return_value = func(*args, **kwargs)
11        end = time.time()
12        print('[*] Время выполнения: {} секунд.'.format(end - start))
13        return return_value
14
15    return wrapper
16
17
18 @benchmark
19 def fetch_webpage(url):
20     import requests
21     webpage = requests.get(url)
22     return webpage.text
23
24
25 ▶ if __name__ == "__main__":
26     webpage = fetch_webpage('https://google.com')
27     print(webpage)
28
```



```
e4 x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe C:/Users/Evil/
[*] Время выполнения: 1.898144245147705 секунд.
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" la
var f=this||self;var h,k=[];function l(a){for(var b;a&&(!a.getAttribute||
function n(a,b,c,d,g){var e="";c||-1!==b.search("&ei=")||(e="&ei="+l(d),-
google.y={};google.sy=[];google.x=function(a,b){if(a)var c=a.id;else{do c
document.documentElement.addEventListener("submit",function(b){var a;if(a
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;
var f=this||self:var n,h,k=null!==(n=f.meil)&&void 0!==(n?n:1).l=null!==(h=f
```

1.1 Индивидуальное задание 1 (рис 1-3)

Вариант 5

Условие: Вводится строка целых чисел через пробел. Напишите функцию, которая преобразовывает эту строку в список чисел и возвращает их сумму. Определите декоратор для этой функции, который имеет один параметр `start` – начальное значение суммы. Примените декоратор со значением `start=5` к функции и вызовите декорированную функцию. Результат отобразите на экране.

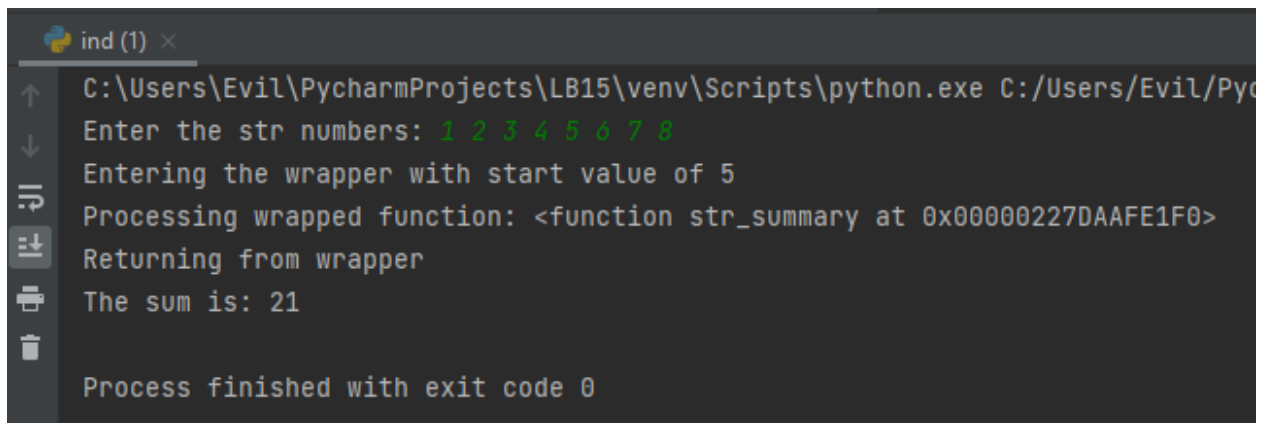
```
ind.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def start_sum(start):
5     def decorator(func):
6         def wrapper(*args):
7             print(f"Entering the wrapper with start value of {start}")
8             print(f"Processing wrapped function: {func}")
9             string = args[0]
10            lst = string.split()
11            lst = lst[start:]
12            start_string = " ".join(lst)
13            return_sum = func(start_string)
14            print("Returning from wrapper")
15            return return_sum
16        return wrapper
17    return decorator
18
19
20 @start_sum(start=5)
21 def str_summary(string):
22     a = [int(s) for s in string.split()]
23     summary = sum(elem for elem in a)
24     return summary
25
26
27 ▶ if __name__ == "__main__":
28     string_inp = input("Enter the str numbers: ")
29     print(f"The sum is: {str_summary(string_inp)}")
30
```

Рисунок 1 – Код программы

```
ind (1) x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe
Enter the str numbers: 1 2 3 4 5 6 7 8
The sum is: 36

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения программы без декоратора



```
ind (1) x
C:\Users\Evil\PycharmProjects\LB15\venv\Scripts\python.exe C:/Users/Evil/Py
Enter the str numbers: 1 2 3 4 5 6 7 8
Entering the wrapper with start value of 5
Processing wrapped function: <function str_summary at 0x00000227DAAFE1F0>
Returning from wrapper
The sum is: 21

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения программы с использованием декоратора

2. Ответы на контрольные вопросы

1. Что такое декоратор?

Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

2. Почему функции являются объектами первого класса?

Потому что с ними можно делать всё то же, что и с любым другим объектом: передавать как параметр, возвращать из функции и присваивать переменной.

3. Каково назначение функций высших порядков?

Функции высших порядков — это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

Они предназначены для передачи функций как параметры другим функциям, а также возвращать их как значения

4. Как работают декораторы?

Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

5. Какова структура декоратора функций?

Объявляется декоратор, который принимает функцию как аргумент, внутри декоратора объявляется обёртка, которая выполняется функцию с изменёнными условиями.

6. Самостоятельно изучить как можно передать параметры декоратору, а не декорируемой функции?

Поместить декоратор внутри функции, которая будет принимать некоторые параметры, а затем использовать эти параметры в декораторе.